DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9729

A MATRIX-GEOMETRIC SOLUTION FOR THE MULTISERVER NONPREEMPTIVE PRIORITY QUEUEING MODEL WITH MIXED PRIORITIES

by

H. LEEMANS

G. DEDENE

Katholieke Universiteit Leuven

Naamsestraat 69, B-3000 Leuven

ONDERZOEKSRAPPORT NR 9729

A MATRIX-GEOMETRIC SOLUTION FOR THE MULTISERVER NONPREEMPTIVE PRIORITY QUEUEING MODEL WITH MIXED PRIORITIES

by

H. LEEMANS

G. DEDENE

D/1997/2376/31

A matrix-geometric solution for the multiserver nonpreemptive priority queueing model with mixed priorities

H. Leemans
K.U.Leuven
Department of Applied Economics
Naamsestraat 69
B-3000 Leuven
Belgium
Fax : +32-16/32.67.32
Phone : +32-16/32.68.90
E-mail : Herlinde.Leemans@econ.kuleuven.ac.be

G. Dedene
K.U.Leuven
Department of Applied Economics
Naamsestraat 69
B-3000 Leuven
Belgium
Fax : +32-16/32.67.32
Phone : +32-16/32.68.75
E-mail : Guido.Dedene@econ.kuleuven.ac.be

September 26, 1997

Abstract

This paper describes the analysis of multiserver queues with nonpreemptive mixed priorities. Such priority structures occur, for example, in initiator settings within the mainframe operating system MVS: job classes have to be assigned to initiators and their priorities may differ amongst the initiators. Results of the analysis provide insight in how average queue lengths in this priority system behave under different class loads. Bounds have to be defined in order to obtain a matrix-geometric solution and it is shown how this affects the average queue lengths. The results should eventually allow to derive guidelines with respect to initiator definitions.

1 Introduction

This paper discusses the analysis of a mixed priority queueing model. The model described has an application in the domain of batch job processing in MVS systems. MVS is IBM's most famous - and complex - mainframe operating system. Its complexity is reflected in the numerous amount of parameters which have to be defined by the performance manager(s) in order to have the system work well. In the following, we will restrict our attention to some parameters required for the scheduling and processing of batch jobs.

Batch jobs in MVS are divided into job classes, based upon their resource requirements (e.g. cpu seconds, number of tape units required, memory requirements, ...). They are executed by MVS in separate batch address spaces, called initiators. The initiator code selects the next job to be processed, loads the job into its address space and passes control to it. After execution, control is passed back to the initiator which cleans up the environment and restarts by selecting a new job. Several initiators may be active at a time, all performing the same functions. This allows multiple batch jobs to be processed in parallel. The number of initiators has to be defined by the system performance manager.

The definition of initiators to the system includes a list of the job classes they are allowed to execute. Single-class initiators are allowed to process jobs of one class only. Multi-class initiators can select jobs of several job classes. A simple initiator definition example is shown in Figure 1. A total of four initiators are defined to be active simultaneously. The first two initiators (I1 and I2) are single-class initiators, executing class A and class B jobs respectively. The next two initiators can process jobs of classes C and D each. The order in which the classes are listed in the initialization code imposes a priority structure on these classes. As such, I3 below selects class C jobs first and takes class D jobs only when the queue of C jobs is empty. Similarly, I4 executes class D jobs before class C jobs. The priorities involved are of a nonpreemptive kind; e.g. a newly arriving D job is not allowed to preempt a class C job being executed by I4.

```
INITDEF   PARTNUM=4
INIT001   CLASS=A, START, NAME=I1
INIT002   CLASS=B, START, NAME=I2
INIT003   CLASS=CD, START, NAME=I3
INIT004   CLASS=DC, START, NAME=I4
```

Figure 1: initiator initialization code : an example

One of the questions arising here is how many initiators are required and how to assign job classes to these initiators. It is therefore important to understand how the class waiting times behave under different initiator settings. For the classes A and B in the example above, this simply amounts to the solution of a single-class single-server queueing model, for which analytic results have been obtained in the literature. The queueing model involved in the calculation of waiting times for classes like C and D above has - to our knowledge - not been studied before.

The remainder of this paper is organized as follows. In Section 2 we will formally describe the queueing model to be analyzed. The matrix-geometric model is outlined in Section 3. We will distinguish two different models, de-

pending on the definition of level and phase. Numerical examples are given in Section 4. Next, Section 5 shows how these particular models may be used to produce upper and lower bounds for the average number of jobs in this priority queueing system. Finally, Section 6 summarizes the main results of the research in this paper. The extension to more than two classes and servers is treated in a forthcoming paper.

2 Formal definition of the model

The queueing model to be analyzed is referred to as a *multiserver nonpreemptive priority queue with mixed priorities.* This should not be confused with so-called mixed priority policies, where the priority of the classes is determined at the beginning of each busy period, based on some parameter(s). In our model, the priorities are fixed with respect to a particular server, but may vary amongst servers. These settings remain the same at any point in time (until they are changed by the performance manager).

For the time being, we assume that the system consists of two servers (initiator I_1 and initiator I_2) and two job classes (A and B) as illustrated in Figure 2. On server I_1, class A has nonpreemptive priority over class B; on server I_2, class B has nonpreemptive priority over class A. Both classes have Poisson arrivals with parameters λ_A and λ_B respectively. Service times are exponentially distributed with average $1/\mu_A$ and $1/\mu_B$. It is generally assumed that μ_A may differ from μ_B. The system can be regarded as having two queues, one for each class. The servers select jobs for service depending on the state of the queues; I_1 will only select a class B job if the class A queue is empty, otherwise it selects class A jobs. I_2 selects class A jobs only if the queue of B jobs is empty, otherwise it selects class B jobs. When both servers are idle, an arriving job is processed by the server on which it has the highest priority. Service discipline within each class is *fcfs.*

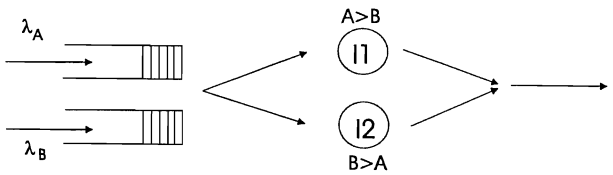

Figure 2: two-class two-server priority queueing model with mixed priorities

Multiserver priority queueing models with Markovian assumptions and two priority classes with unequal average service times have been studied before [1, 2, 3, 7, 8]. This particular queueing model however differs from the previously studied models in that the priorities also depend on the servers.

3 Matrix-geometric models

We denote the state of the system by the tuple (i, j, x, y), where i and j respectively represent the number of class A and class B jobs in the system (in the queue or in service). As such, i and j can take the integer values 0,1,2,....

The indices x and y refer to the class of job that is being served on I_1 and I_2 respectively. Consequently, their values may be A, B or 0; the latter indicates that the respective server is idle. It is necessary to include this information in the state description. Since the classes may have unequal service times, the class that is being served determines the time at which a particular server becomes idle and as such the class that is served next.

We will define two possible structures for the generator matrix Q, depending on how the level and phase of the process are chosen:

1. Model 1 (Class Bounded Model, CBM):
   - Level = the number of class A jobs (i)
   - Phase = the number of class B jobs (j)
2. Model 2 (Difference Bounded Model, DBM):
   - Level = the minimum of i and j
   - Phase = the difference $i - j$

The indices x and y will be included as a "phase-within-the-phase"; (x, y) is therefore called the *minor* phase as opposed to the *major* phase j. In order to obtain finite block matrices, it is necessary to limit the number of major phases in both models. In the CBM case, we will therefore define an upper bound M on the number of B jobs; for DBM, M represents the absolute value of the maximal allowable difference between the two queues. In the following paragraphs, both models are generally described. More details on the structure of the matrices may be found in the appendix.

3.1 Class Bounded Model

3.1.1 The generator matrix

Each level of the CBM model consists of $M+1$ major phases. Also, each major phase consists of a number of minor phases, depending on the level i and on the major phase j. It is easy to see that major phase 1 within level 1 has only two minor phases, (A, B) and (B, A), whereas the same major phase within level 2 has three minor phases, (A, A),(A, B) and (B, A). The boundary levels 0 and 1 will therefore be slightly different from the other levels. The minor phases for all levels are summarized in Table 1.

	major phase = 0	major phase = 1	major phase $\geq$ 2
level 0	(0,0)	(0,B),(B,0)	(B,B)
level 1	(A,0),(0,A)	(A,B),(B,A)	(A,B),(B,A),(B,B)
level $\geq$ 2	(A,A)	(A,A),(A,B),(B,A)	(A,A),(A,B),(B,A),(B,B)

Table 1: minor phases for each major phase in each level (CBM)

The generator matrix Q for this model has the following structure:

$$Q = \begin{bmatrix} B_{00} & B_{01} & & & & \\ B_{10} & B_{11} & B_{12} & & & \\ & B_{21} & A_1 & A_0 & & \\ & & A_2 & A_1 & A_0 & \\ & & & A_2 & A_1 & \ddots \\ & & & & A_2 & \ddots \\ & & & & & \ddots \end{bmatrix} \tag{1}$$

Transitions occur between two levels or between two major phases within the same level, with a possible transition in the minor phase at the same time. The behavior of the system "away from the boundary" is described in Table 2 and Table 3. When the minor phase is indicated by an asterisk (*), the transition preserves the minor phase of the state that is left (e.g. (B, A) remains (B, A) after the transition).

	from state	$\rightarrow$	to state	rate	condition
level up	$(i, j, *, *)$	$\rightarrow$	$(i+1, j, *, *)$	λ_A	
level down	(i, j, A, B)	$\rightarrow$	$(i-1, j, A, B)$	μ_A	
	(i, j, A, A)	$\rightarrow$	$(i-1, j, A, A)$	$2\mu_A$	phase $= 0$
		$\rightarrow$	$(i-1, j, A, A)$	μ_A	phase > 0
		$\rightarrow$	$(i-1, j, A, B)$	μ_A	phase > 0
	(i, j, B, A)	$\rightarrow$	$(i-1, j, B, A)$	μ_A	phase $= 1$
		$\rightarrow$	$(i-1, j, B, B)$	μ_A	phase > 1

Table 2: transition table for CBM : changes between levels

	from state	$\rightarrow$	to state	rate	condition
phase up	$(i, j, *, *)$	$\rightarrow$	$(i, j+1, *, *)$	λ_B	
phase down	(i, j, A, B)	$\rightarrow$	$(i, j-1, A, A)$	μ_B	phase $= 1$
		$\rightarrow$	$(i, j-1, A, B)$	μ_B	phase > 1
	(i, j, B, A)	$\rightarrow$	$(i, j-1, A, A)$	μ_B	
	(i, j, B, B)	$\rightarrow$	$(i, j-1, A, B)$	μ_B	
		$\rightarrow$	$(i, j-1, B, A)$	μ_B	phase $= 2$
		$\rightarrow$	$(i, j-1, B, B)$	μ_B	phase > 2

Table 3: transition table for CBM : changes within the level

These tables allow us to define the matrices A_0, A_1 and A_2 as follows :

$$A_0 = \begin{bmatrix} L_{A0} & & & & & \\ & L_{A1} & & & & \\ & & L_A & & & \\ & & & L_A & & \\ & & & & L_A & \\ & & & & & \ddots \end{bmatrix},$$

$$A_1 = \begin{bmatrix} D_0 & L_{B0} & & & & \\ M_{B0} & D_1 & L_{B1} & & & \\ & M_{B1} & D & L_B & & \\ & & M_B & \ddots & \ddots & \\ & & & \ddots & D & L_B \\ & & & & M_B & D_M \end{bmatrix}$$

$$\text{and } A_2 = \begin{bmatrix} M_{A0} & & & & & \\ & M_{A1} & & & & \\ & & M_A & & & \\ & & & M_A & & \\ & & & & \ddots & \end{bmatrix}.$$

The elements of A_0, A_1 and A_2 are blocks; their structure is displayed in the appendix. The structure of the boundary matrices is more messy; it may however easily be derived and is therefore not shown here.

3.1.2 Ergodicity constraint

An irreducible QBD with irreducible matrix A and with a finite number of phases is positive recurrent if and only if $\gamma A_2 \mathbf{1} > \gamma A_0 \mathbf{1}$, where γ is the stationary probability vector of A and $\mathbf{1}$ is an appropriately sized column vector of ones (simple drift condition, theorem 7.2.1 in Latouche and Ramaswami [5] and theorem 1.3.2 in Neuts [9]). The QBD represented by CBM is clearly irreducible. However, the matrix $A = A_0 + A_1 + A_2$ is reducible. This is easily seen from the dynamics of the system "away from the boundary". From Table 2 and Table 3, it is clear that no transition is possible from states with minor phases (A, A) or (A, B) into states with minor phases (B, A) or (B, B). The latter states may be reached only by passing through the boundary and, once they are left at the higher levels, the process cannot return into these states until it reaches the boundary levels.

As such, the reducible matrix A has one unique irreducible class of phases, containing only the states with minor phase (A, A) and (A, B). By permuting the appropriate rows and columns, redefine the matrix A as

$$A = \begin{bmatrix} K^{(1)} & 0 \\ L^{(1)} & L^{(0)} \end{bmatrix}$$

where $K^{(1)}$ is irreducible and contains only the states with minor phases (A, A) and (A, B). The matrices A_i are similarly structured, such that

$$A_i = \begin{bmatrix} K_i^{(1)} & 0 \\ L_i^{(1)} & L_i^{(0)} \end{bmatrix}$$

Now, applying theorem 7.3.1 from Latouche and Ramaswami [5], we may state that the CBM model is positive recurrent if and only if $\gamma^{(1)} K_2^{(1)} \mathbf{1} >$

$\gamma^{(1)} K_0^{(1)} \mathbf{1}$, where $\gamma^{(1)}$ is the stationary probability vector of $K^{(1)}$. This reduces to the condition

$$\lambda_A \leq \mu_A \left[1 + \sum_{j=0}^{M} \gamma_j^{(AA)} \right] \tag{2}$$

where $\gamma_j^{(AA)}$ $(j = 1, \ldots, M)$ are the elements of the stationary probability vector $\gamma^{(1)}$ corresponding to the states with major phase j and minor phase (A, A).

3.2 Difference Bounded Model

3.2.1 The generator matrix

In this model, the level is defined as $\min(i, j)$, the phase is $i-j$ (may be negative) and M is the maximal allowable value of $|i - j|$. Each level consists of $2M + 1$ major phases. As with CBM, each major phase consists of a number of minor phases, depending on the level i and on the major phase j. The minor phases for all levels are summarized in Table 4.

	major phase	minor phases
level 0	< -1	(B,B)
	$= -1$	$(0,B),(B,0)$
	$= 0$	$(0,0)$
	$= +1$	$(A,0),(0,A)$
	$> +1$	(A,A)
level 1	< 0	$(A,B),(B,A),(B,B)$
	$= 0$	$(A,B),(B,A)$
	> 0	$(A,A),(A,B),(B,A)$
level ≥ 2	all phases	$(A,A),(A,B),(B,A),(B,B)$

Table 4: minor phases for each major phase in each level (DBM)

The generator matrix Q has the same form as (1). In order to construct the matrices A_i, we summarize the behavior of the system "away from the boundary" in Table 5 and Table 6.

	from state	$\rightarrow$	to state	rate	condition
level up	$(i,j,*,*)$	$\rightarrow$	$(i+1,j+1,*,*)$	λ_A	phase < 0
		$\rightarrow$	$(i+1,j-1,*,*)$	λ_B	phase > 0
level down	(i,j,A,A)	$\rightarrow$	$(i-1,j-1,A,A)$	μ_A	$-M <$ phase ≤ 0
		$\rightarrow$	$(i-1,j-1,A,B)$	μ_A	$-M <$ phase ≤ 0
	(i,j,A,B)	$\rightarrow$	$(i-1,j-1,A,B)$	μ_A	$-M <$ phase ≤ 0
		$\rightarrow$	$(i-1,j+1,A,B)$	μ_B	$0 \leq$ phase $< +M$
	(i,j,B,A)	$\rightarrow$	$(i-1,j-1,B,B)$	μ_A	$-M <$ phase ≤ 0
		$\rightarrow$	$(i-1,j+1,A,A)$	μ_B	$0 \leq$ phase $< +M$
	(i,j,B,B)	$\rightarrow$	$(i-1,j+1,A,B)$	μ_B	$0 \leq$ phase $< +M$
		$\rightarrow$	$(i-1,j+1,B,B)$	μ_B	$0 \leq$ phase $< +M$

Table 5: transition table for DBM : changes between levels

We may now define the matrices A_0, A_1 and A_2 as

	from state	$\rightarrow$	to state	rate	condition
phase up	$(i,j,*,*)$	$\rightarrow$	$(i,j+1,*,*)$	λ_A	$0 \leq$ phase $< +M$
	(i,j,A,B)	$\rightarrow$	$(i,j+1,A,B)$	μ_B	phase < 0
	(i,j,B,A)	$\rightarrow$	$(i,j+1,A,A)$	μ_B	phase < 0
	(i,j,B,B)	$\rightarrow$	$(i,j+1,A,B)$	μ_B	phase < 0
		$\rightarrow$	$(i,j+1,B,B)$	μ_B	phase < 0
phase down	$(i,j,*,*)$	$\rightarrow$	$(i,j-1,*,*)$	λ_B	$-M <$ phase ≤ 0
	(i,j,A,A)	$\rightarrow$	$(i,j-1,A,A)$	μ_A	phase > 0
		$\rightarrow$	$(i,j-1,A,B)$	μ_A	phase > 0
	(i,j,A,B)	$\rightarrow$	$(i,j-1,A,B)$	μ_A	phase > 0
	(i,j,B,A)	$\rightarrow$	$(i,j-1,B,B)$	μ_A	phase > 0

Table 6: transition table for DBM : changes within the level

$$A_0 = \begin{bmatrix} \mathbf{0} & L_A & & & & & \\ & \ddots & & & & & \\ & & L_A & & & & \\ & & \mathbf{0} & & & & \\ & & L_B & & & & \\ & & & \ddots & & & \\ & & & & L_B & \mathbf{0} \end{bmatrix},$$

$$A_1 = \begin{bmatrix} D_{-M} & M_B & & & & & \\ L_B & D & \ddots & & & & \\ & \ddots & \ddots & M_B & & & \\ & & L_B & D & L_A & & \\ & & & M_A & \ddots & \ddots & \\ & & & & \ddots & D & L_A \\ & & & & & M_A & D_{+M} \end{bmatrix}$$

$$\text{and } A_2 = \begin{bmatrix} \mathbf{0} & & & & & & \\ M_A & & & & & & \\ & \ddots & & & & & \\ & & M_A & \mathbf{0} & M_B & & \\ & & & & & \ddots & \\ & & & & & & M_B \\ & & & & & & \mathbf{0} \end{bmatrix}$$

where $\mathbf{0}$ represents a 4×4 block of zeros.

3.2.2 Ergodicity constraint

As with CBM, the matrix A is reducible. It is clear from Table 5 and Table 6 that no transition is possible from states with minor phase (A,B) into states with minor phases (A,A),(B,A) or (B,B). Here, the reducible matrix A has one unique irreducible class of phases containing only the states with minor phase (A,B). Following the same arguments as in Section 3.1.2, we may state that the QBD is positive recurrent if and only if

$$X\left(\mu_A + X^{M-1}\mu_B\right) > \lambda_A + X^{M+1}\lambda_B \tag{3}$$

where

$$X = \frac{\lambda_A + \mu_B}{\lambda_B + \mu_A} \tag{4}$$

Remark: The stability condition is formulated in terms of the individual values of λ and μ and of the bound M. For the original unbounded model, we know (intuitively) that the ergodicity condition for a two-class M/M/2 applies, simply requiring

$$\rho_A + \rho_B < 2 \tag{5}$$

If $M \to \infty$, the number of phases $\to \infty$. Theorem 7.2.1 of [5] is no longer valid and it is impossible to derive (5) from the drift condition above (which results in $\rho_A \rho_B < 1$ for $M \to \infty$).

4 Numerical examples

Using numerical examples, we may analyze the behavior of the mixed priority queues and the influence of the bound on the results of the models. Both CBM and DBM models are implemented in Matlab. The rate matrix R is calculated with the logarithmic reduction algorithm (the algorithm Ex; see Latouche and Ramaswami [4]).

Next, the boundary probability vectors ($\mathbf{y}_0$, $\mathbf{y}_1$ and $\mathbf{y}_2$) are defined by solving

$$\begin{aligned}
\mathbf{y}_0 B_{00} + \mathbf{y}_1 B_{10} &= \mathbf{0} && (6)\\
\mathbf{y}_0 B_{01} + \mathbf{y}_1 B_{11} + \mathbf{y}_2 B_{21} &= \mathbf{0} && (7)\\
\mathbf{y}_1 B_{12} + \mathbf{y}_2 (A_1 + R A_2) &= \mathbf{0} && (8)\\
\mathbf{y}_0 \mathbf{1} + \mathbf{y}_1 \mathbf{1} + \mathbf{y}_2 (I - R)^{-1} \mathbf{1} &= 1 && (9)
\end{aligned}$$

Then, using R and the boundary probability vectors, the first and second moments of the marginal queue length distribution are easily calculated.

We assume that $\mu_A = \mu_B = 1$ (a more extensive set of results is available with the authors). The load of the system is varied by choosing appropriate values of λ_A and λ_B. It is clear that the models, because of the finite value of M, produce results which are only an approximation for the unbounded system. For one set of data, we obtained 'exact' results from CBM by increasing the value of M until the average queue lengths (from now on denoted by N_A and N_B) were stable. For the case where $\lambda_A = \lambda_B = 0.95$, the value of M had to be pushed over 200 in order to have a stable average queue length for both classes. In the paragraphs below, we only show results for high class A load and varying class B load, because the average queue lengths are most strongly affected by M in these cases.

4.1 Results for CBM

It is easy to see that CBM produces a lower bound for the average queue lengths of both classes. When the bound M on the number of class B jobs is reached, new class B arrivals are immediately rejected from the system. The effective load of class B is reduced and the average queue length of B jobs will therefore be smaller than in the unbounded system (primary effect). As a secondary effect, because there are less B jobs in the system, a larger part of the capacity is left for class A jobs and they will be served faster, as such reducing class A average queue length. The lower M and the higher the load on class B, the more jobs will be rejected from the system and the larger the impact will be on the average queue lengths. This is clearly seen in figures 3 and 4. They represent the average queue length of class A and B respectively, for a system with class A load $= 0.95$, and class B load varying up to 0.95. The results improve with higher values of M. With $M = 40$, N_A is already close to the 'exact' results. N_B is less accurate for the same values of M. We may state that CBM performs well if the classes have different loads and if the level of the process is chosen to represent the class with the highest load.

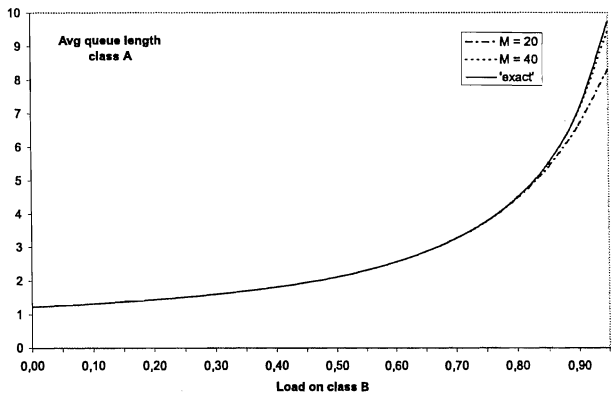

Figure 3: Average queue length for class A when $\rho_A = 0.95$

4.2 Results for DBM

For DBM, the effect of M on the average queue lengths is more complex. The bound M may be exceeded by different events, depending on the phase of the process. E.g. when the phase is positive and equal to M, thus $i - j = +M$, then the next arrival of an A job (occurring at rate λ_A) causes the limit to be exceeded and will therefore be rejected from the system. This primarily affects N_A by decreasing its value. However, with positive phase, M may also be exceeded by a completion of a B job (occurring at rate μ_B). This B job is then forced to remain in service until $i-j < M$, as such increasing N_B. Table 7 summarizes these effects.

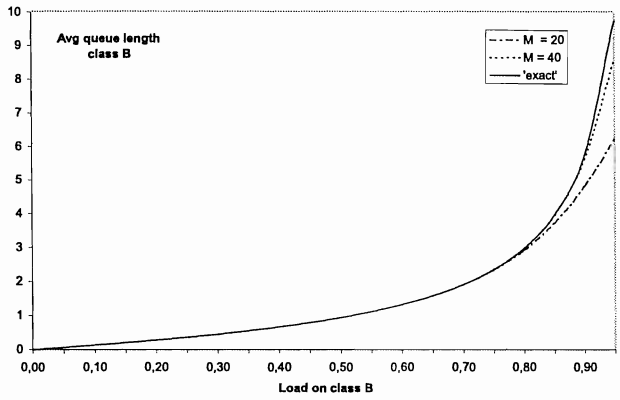

Figure 4: Average queue length for class B when $\rho_A = 0.95$

	M exceeded by	primary effect
phase +	λ_A	$N_A \searrow$
	μ_B	$N_B \nearrow$
phase -	λ_B	$N_B \searrow$
	μ_A	$N_A \nearrow$

Table 7: Effects of M on the average queue lengths

Now, distinguish the following three cases :

1. $\rho_A > \rho_B \Rightarrow \text{Prob(phase} = \text{pos)} > \text{Prob(phase} = \text{neg)}$
2. $\rho_A < \rho_B \Rightarrow \text{Prob(phase} = \text{pos)} < \text{Prob(phase} = \text{neg)}$
3. $\rho_A = \rho_B \Rightarrow \text{Prob(phase} = \text{pos)} = \text{Prob(phase} = \text{neg)}$

For case 1, with $\rho_A > \rho_B$, the primary effect of M is to decrease the value of N_A and to increase the value of N_B, as such producing a lower bound for the average queue length in class A and an upper bound for the average queue length in class B. The opposite happens in case 2, resulting in an upper bound for class A and a lower bound for class B. In case 3, positive and negative phases have equal probability. Under the assumption that $\lambda_A < \mu_A$ and $\lambda_B < \mu_B$, it is experimentally observed that DBM produces an upper bound for the queue lengths of both classes. These conclusions are summarized in Table 8.

$\rho_A > \rho_B \Rightarrow$ Prob(phase +) > Prob(phase -) $\Rightarrow$	lower bound for class A upper bound for class B
$\rho_A < \rho_B \Rightarrow$ Prob(phase +) < Prob(phase -) $\Rightarrow$	upper bound for class A lower bound for class B
$\rho_A = \rho_B \Rightarrow$ Prob(phase +) = Prob(phase -) $\Rightarrow$	upper bound for both classes

Table 8: Bounds produced by DBM

Figures 5 and 6 show, for $\rho_A = 0.95$, the difference between the average queue lengths produced by DBM (for $M = 10$ and $M = 20$) and the 'exact' results, expressed in percentage of 'exact'. It is clear how DBM produces lower bounds for N_A as long as $\rho_B < \rho_A (= 0.95)$, while we obtain an overall upper bound for N_B. These bounds are not monotonically increasing with the load and are therefore not practically useful. In the following section, we will modify this model in order to obtain upper and lower bounds for the average queue lengths for the whole set of data.

5 Upper and lower bounds

We have shown in section 4.1 how CBM produces lower bounds for the average queue lengths of the two-class two-server mixed priority model. It is however far more interesting to have both upper and lower bounds. In this section, we modify part of the model (for CBM as well as for DBM) for that purpose. Similar ideas have been applied earlier in other queueing models (e.g. [6, 10]).

5.1 Bounds for CBM

We make the following assumption: *if M is reached, a newly arriving B job is added to the A queue and is further treated as an A job.*

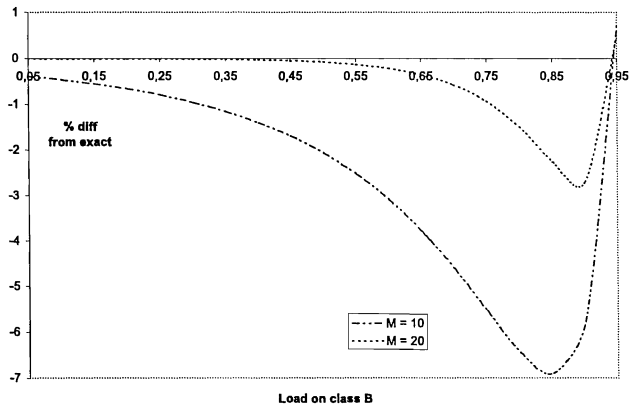

Figure 5: % deviation for class A when $\rho_A = 0.95$

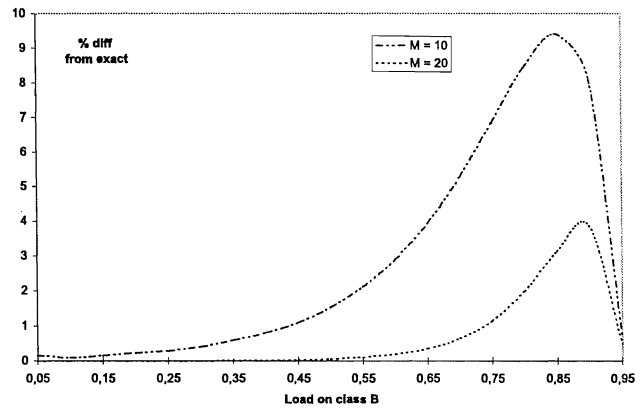

Figure 6: % deviation for class B when $\rho_A = 0.95$

With this modification, the overall system load is maintained. It results in a lower bound for N_B because a number of B jobs will not join their own queue. On the other hand, this assumption increases the number of jobs in the A queue and produces as such an upper bound for N_A. A lower bound for N_A and an upper bound for N_B are obtained by simply switching the classes (making class B the level and class A the phase).

This assumption requires only a slight modification to the matrices representing 'level up' transitions: B_{01}, B_{12} and A_0. When $j = M$, a transition to the next higher level is not only caused by the arrival of an A job, but also by the arrival of a B job (which is transformed into an A). The transition rate for these states changes from λ_A into $\lambda_A + \lambda_B$. A_0 now becomes

$$A_0 = \begin{bmatrix} L_{A0} & & & & & \\ & L_{A1} & & & & \\ & & L_A & & & \\ & & & L_A & & \\ & & & & \ddots & \\ & & & & & L_A^U \end{bmatrix},$$

where L_{A0}, L_{A1} and L_A are defined as before (Appendix, CBM), and L_A^U replaces the block corresponding to major phase M with $(\lambda_A + \lambda_B)I_4$. The modifications to the other matrices are similar.

The upper and lower bounds for the same set of data as in Section 4 are shown in Figures 7 through 10. Figures 7 and 8 show the results for class A with $M = 20$ and $M = 40$ respectively. The lower bound defined in this section is less tight than the upper bound and also less tight than the lower bound resulting from the original CBM. The main reason for this is that the new lower bound is generated by the reversed model in which class A represents the phase of the process. Because ρ_A is chosen to be 0.95, the influence of M is quite strong. The lower bound obtained in the previous section in conjunction with the upper bound derived here gives a good impression of the range within which the exact solution will be. This range is already reasonably narrow for $M = 40$.

Similar arguments can be used to explain the results in Figures 9 and 10. The lower bounds are almost coinciding; the upper bound is less tight. Again, the upper bound for class B is now generated by the reversed model (with class A being the phase) and is strongly influenced by M because of the high class A load. The results improve greatly with $M = 40$.

5.2 Bounds for DBM

We make the following assumption: *if M is exceeded by the arrival of an A (B) job, a new B (A) job is added to the system.*

This assumption generates upper bounds for the average queue lengths of both classes. When an arriving job causes the bound to be exceeded, not only is it accepted in the queue, it also increases the queue length of the other class by creating a new job there. Only 'level up' matrices are affected: B_{01}, B_{12} and A_0. The latter becomes

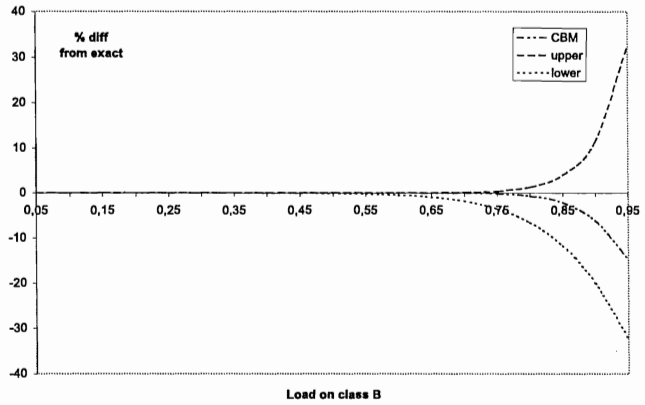

Figure 7: % deviation between bounds and 'exact' N_A, $\rho_A = 0.95$, $M = 20$

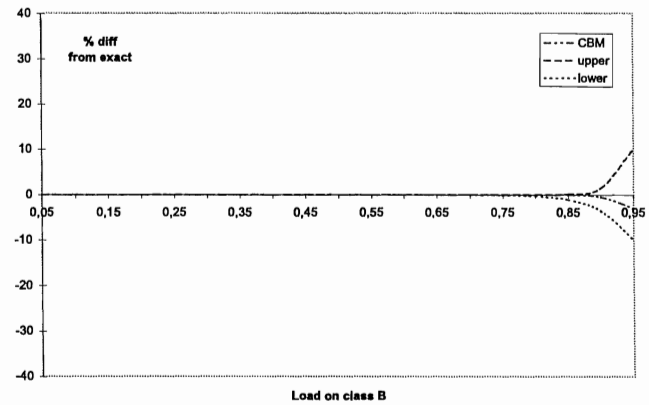

Figure 8: % deviation between bounds and 'exact' N_A, $\rho_A = 0.95$, $M = 40$

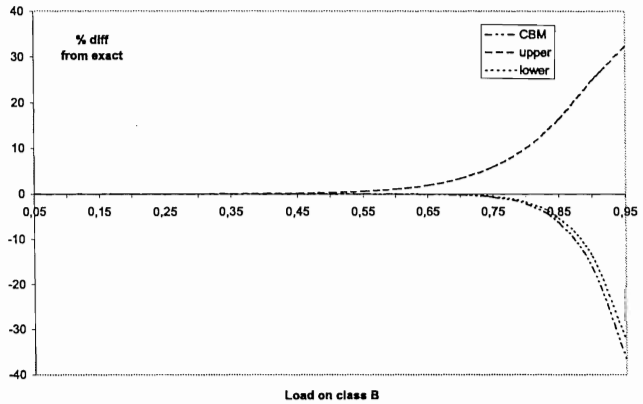

Figure 9: % deviation between bounds and 'exact' N_B, $\rho_A = 0.95$, $M = 20$

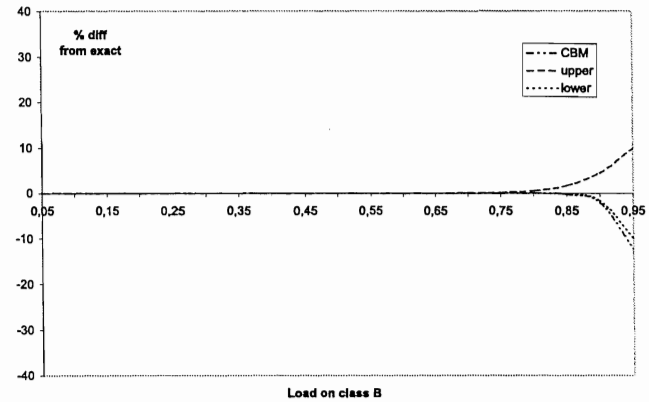

Figure 10: % deviation between bounds and 'exact' N_B, $\rho_A = 0.95$, $M = 40$

$$A_0 = \begin{bmatrix} L_B & L_A & & & & & \\ & \ddots & & & & & \\ & & L_A & & & & \\ & & 0 & & & & \\ & & L_B & & & & \\ & & & \ddots & & & \\ & & & & L_B & L_A \end{bmatrix}.$$

The modifications to B_{01} and B_{12} are similar.

Generating lower bounds requires the following assumption (applied to the original DBM): *if M is exceeded by the completion of an A (B) job, an extra B (A) job is removed from the system.*

It is immediately clear that this assumption reduces the average queue lengths of both classes. It requires slight modifications to the 'level down' matrices: B_{10}, B_{21} and A_2. A_2 now becomes

$$A_2 = \begin{bmatrix} M_A & & & & & & \\ M_A & & & & & & \\ & \ddots & & & & & \\ & & M_A & 0 & M_B & & \\ & & & & & \ddots & \\ & & & & & & M_B \\ & & & & & & M_B \end{bmatrix}$$

and the modifications to the other matrices are similar.

Figures 11 and 12 show the behavior of the bounds for class A and B respectively. The bounds are monotonically increasing with the load of the system, but their accuracy for low values of M is rather poor. The lower bound is only reasonably close for $M = 40$. The upper bound is even worse.

The reason for this behavior of the upper bound is mainly the fact that the modification increases the total load of the system. For the highest class loads, the system is even no longer stable. From the definition of the matrices A_i, it is clear that the modification does not lower the phase of the process. Suppose that the process is in state $(i, +M)$. In order to obtain upper bounds, the arrival of an A job adds an additional B job to the system and puts the process in state $(i+1, +M)$. In these phases, the process thus moves level up at rates $\lambda_A + \lambda_B$. All following arrivals of A have the same effect, unless the arrival of a B job or the completion of an A job changes the phase. With low values of M and high class loads, this results in an exponentially increasing value for N_A and N_B. Similar arguments may explain the behavior of the lower bound. The modification decreases the total load of the system. Of course, the higher the class loads and the lower the value of M, the higher the load reduction will be.

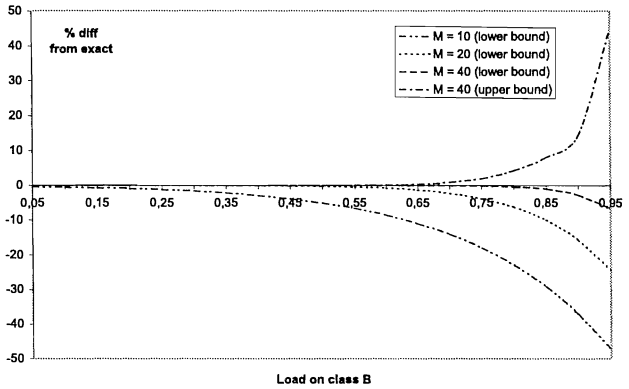

Figure 11: % deviation between bounds and 'exact' N_A, $\rho_A = 0.95$

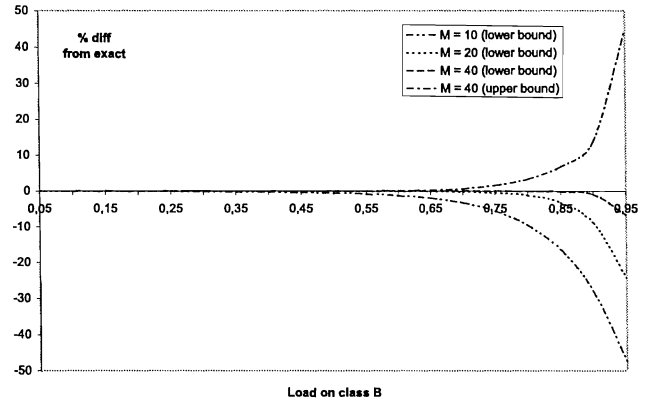

Figure 12: % deviation between bounds and 'exact' N_B, $\rho_A = 0.95$

6 Concluding remarks

The results presented in this paper clearly show how the average queue lengths are affected by the use of bounds. CBM results in lower bounds for both classes, which are reasonably accurate and achievable with an acceptable computational effort. This model performs best when the class loads are unequal and when the level represents the class with the highest load. The bounds produced with DBM are quite accurate if both class loads are equal. For such cases, DBM generates upper bounds for both classes.

Overall, we may state that CBM is more practical to work with than DBM. A major reason is the fact that the upper and lower bounds, derived in Section 4.1, give a reasonably good idea of the range within which the exact solution will be found. This is not the case with the upper and lower bounds derived from DBM.

Acknowledgement

The authors would like to thank Guy Latouche (Université Libre de Bruxelles, Belgium) for his invaluable help in this research.

References

[1] H. R. Gail, S. L. Hantler, and B. A. Taylor. Analysis of a non-preemptive priority multiserver queue. *Adv. Appl. Prob.*, 20:852–879, 1988.

[2] H. R. Gail, S. L. Hantler, and B. A. Taylor. On a preemptive Markovian queue with multiple servers and two priority classes. *Mathematics of Operations Research*, 20:365–391, 1988.

[3] E. P. C. Kao and K. S. Narayanan. Modeling a multiprocessor system with preemptive priorities. *Management Science*, 37(2):185–197, 1991.

[4] G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for Quasi-Birth-Death processes. *J. Appl. Prob.*, 30:650–674, 1993.

[5] G. Latouche and V. Ramaswami. Introduction to Matrix Analytic Methods in Stochastic Modeling. To be published, 1997.

[6] J. C. S. Lui, R. R. Muntz, and D. Towsley. Bounding the Mean Response Time of the Minimum Expected Delay Routing Policy: An Algorithmic Approach. *IEEE Transactions on Computers*, 44:1371–1382, 1995.

[7] D. R. Miller. Steady-state algorithmic analysis of M/M/c two-priority queues with heterogeneous servers. In R. L. Disney and T. J. Ott, editors, *Applied Probability - Computer Science, The Interface*, volume II, pages 207–222. Birkhaüser, Boston, 1992.

[8] I. Mitrani and P. J. B. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1:118–125, 1981.

[9] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach.* The John Hopkins University Press, Baltimore, Md., 1981.

[10] D. Towsley and S. Chen. Design and Evaluation of Scheduling Policies for Two Server Fork/Join Queueing Systems. Technical Report COINS TR91-39, University of Massachusetts (Amherst), Department of Computer and Information Science, 1991.

Appendix

JBM

The submatrices of A_0, A_1 and A_2 are defined as follows:

$$L_{A0} = \lambda_A, \qquad L_{A1} = \lambda_A I_3, \qquad L_A = \lambda_A I_4,$$

$$L_{B0} = \begin{bmatrix} & \lambda_B & \end{bmatrix}, \quad L_{B1} = \begin{bmatrix} \lambda_B & & \\ & \lambda_B & \\ & & \lambda_B \end{bmatrix}, \quad L_B = \lambda_B I_4,$$

$$M_{A0} = 2\mu_A, \qquad M_{A1} = \begin{bmatrix} \mu_A & \mu_A & \\ & \mu_A & \\ & & \mu_A \end{bmatrix}, \quad M_A = \begin{bmatrix} \mu_A & \mu_A & \\ & \mu_A & \\ & & \mu_A \end{bmatrix},$$

$$M_{B0} = \begin{bmatrix} \mu_B \\ \mu_B \end{bmatrix}, \qquad M_{B1} = \begin{bmatrix} & \mu_B & \\ \mu_B & & \\ & \mu_B & \mu_B \end{bmatrix}, \quad M_B = \begin{bmatrix} & \mu_B & \\ \mu_B & & \\ & \mu_B & \mu_B \end{bmatrix}.$$

Within each block, the minor phases are ordered as in table 1. I_i denotes an identity matrix of size i. The matrices D_0, D_1, D and D_M are diagonal matrices, the elements of which ensure that the row sums of Q equal zero.

DBM

The submatrices of A_0, A_1 and A_2 are defined as follows:

$$L_A = \lambda_A I_4, \qquad L_B = \lambda_B I_4,$$

$$M_A = \begin{bmatrix} \mu_A & \mu_A & & \\ & \mu_A & & \\ & & \mu_A & \\ & & & \end{bmatrix}, \quad M_B = \begin{bmatrix} & \mu_B & & \\ \mu_B & & & \\ & \mu_B & \mu_B & \end{bmatrix}.$$

I_i denotes an identity matrix of size i. The matrices D_{-M}, D and D_{+M} are diagonal matrices; their elements ensure that the row sums of Q equal zero.